1964

Digital control of a second order hybrid system

McCullough, Lawrence E.; Nash, Gordon C. Jr.

Monterey, California: U.S. Naval Postgraduate School

DIGITAL CONTROL OF A SECOND ORDER HYBRID SYSTEM

LAWRENCE E. McCULLOUGH
GORDON C. NASH, JR.

DIGITAL CONTROL OF A SECOND ORDER HYBRID SYSTEM

by

Lawrence E. McCullough
Lieutenant, United States Navy

and

Gordon C. Nash, Jr.
Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

United States Naval Postgraduate School
Monterey, California

1 9 6 4

DIGITAL CONTROL OF A SECOND ORDER HYBRID SYSTEM

by

Lawrence E. McCullough

and

Gordon C. Nash, Jr.

This work is accepted as fulfilling
the thesis requirements for the degree of
MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING
from the
United States Naval Postgraduate School

ABSTRACT

This paper presents the results of efforts to implement a hybrid control system at the U. S. Naval Postgraduate School. The salient features of this effort are use of real time digital computor programming, digital determination of instantaneous state variables, and, using the foregoing features to demonstrate optimum control of a second order system. It was found that the techniques employed would control the second order system, and that the derivative of a variable could be determined with the digital equipment.

The computer programs for accomplishing this effort are included in Chapter IV and the appendicies.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

APPENDICIES

CHAPTER I

INTRODUCTION

The purpose of this work was to implement a hybrid control system at the U. S. Naval Postgraduate School.

The control problem was approached assuming that a prospective control policy would require instantaneous knowledge of state variables which could not be obtained from the analog sub-system. States needed to implement the control policy would therefore be calculated by the digital computor.

In order to achieve control and obtain the states (i.e., derivatives) it was necessary to be able to determine with considerable accuracy the times required to sample and to compute. Also, appropriate constant time floating point arithmetic and converting procedures had to be developed.

In addition, it was desired to determine what effects on state determination and control any noise present in the system might have.

CHAPTER II

EQUIPMENT

The system under consideration is shown in Fig. 1. The digital computer was a Control Data Corporation 160. The memory capacity is 4096 twelve bit binary word with a storage cycle time of 6.4 microseconds. Operation is controlled by a program internally stored in sequential locations. More complete information including a listing of computer instructions is contained in Ref. 1.

The plant was simulated on a PACE TR 10 Electronic Associates Incorporated analog computer. It is a twenty amplifier computer with a saturation level of $\pm$ 10 volts. The power supply is regulated to $\pm$ 10 millivolts. Further information may be found in Ref. 2.

The A/D and D/A converters were designed and constructed at the U. S. Naval Postgraduate School using Digital Equipment Corporation modules. The principle of operation is successive approximation conversion. The analog capacity was 0 to -10 volts with a sensitivity of 2.4 millivolts per bit. The digital ranges 0000 to 3777 and 4000 to 7777 correspond to analog ranges of -5 to 0 and -10 to -5 volts respectively.

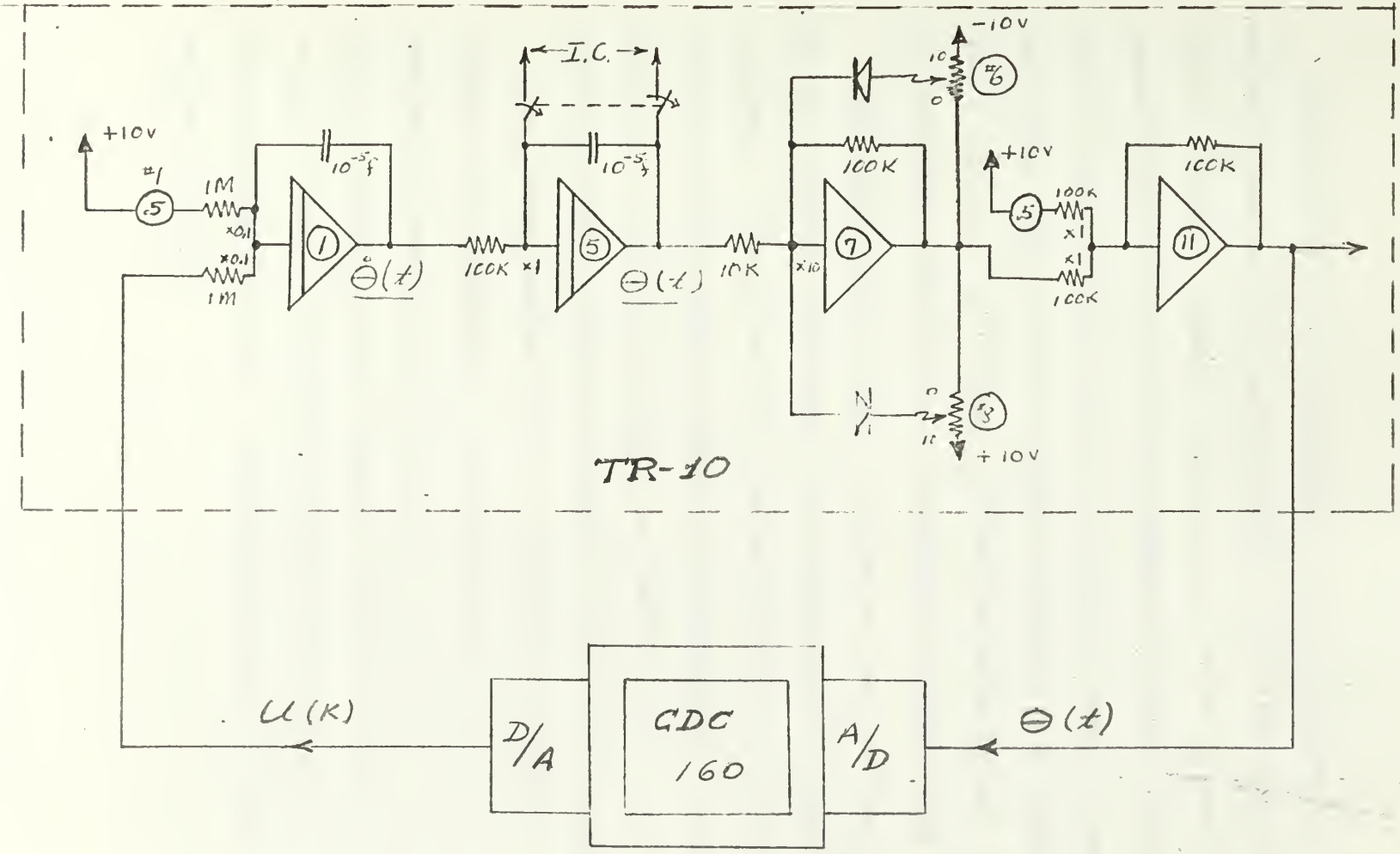


FIGURE 1

HYBRID CONTROL SYSTEM

CHAPTER III

DIGITAL TIME SCALING

Of the factors to be considered in the implementation of a hybrid control system one of the more important is the time scaling of the digital computer. Normally, time is not one of the criterion of digital computer programming, but when the computer is used as part of a control system, it must be time scaled to be compatable with the remainder of the system. Often this time scaling is referred to as computing in "real" time; that is, the time scale of the analog system. The following discussion presents some of the aspects of digital time scaling.

All digital programs must be written so that the time required for the program is independent of the magnitude of any input or computed variables. In many cases this involves padding the shorter paths to equalize all path times to that of the longest path in a particular operation. As a note of caution, care should be exercised to avoid a prohibitive delay which would negate the value of the program.

Any library routine used in a compensation program must meet this constant time requirement. In most cases this was not one of the original requirements, meaning the routine must be rewritten. This has been done for the floating point add, subtract, multiply and divide and the fixed-to-floating point and floating-to-fixed point conversion routines. These are presented in App. I as the Constant Time Floating Point Arithmetic Package.

In writing constant time programs for the CDC 160 the cycle time was used as a basis. Each of the CDC 160 instructions requires one to three cycle times for execution. By writing programs so that all possible paths contain the same number of cycles the constant time requirement was satisfied. Knowing the cycle time, the program time was calculated.

Although Ref. 1 states the cycle time for the CDC 160 is 6.4 microseconds, it was found that the actual time may vary slightly from that value. In the case of programs which contain several hundred cycles the difference may become significant. By timing the below program, which contains 20×10^6 cycles, with a stopwatch the cycle time may be determined.

```
1000   2600   lcc 00      1010   2200   ldc 00
1001   2341   2341        1011   4512   4512
1002   4213   stf 13      1012   0701   sbn 01
1003   2600   lcc 00      1013   6501   nzb 01
1004   7640   7640        1014   7700   hlt
1005   4211   stf 11      1015   0000
1006   5610   aof 10      1016   0000
1007   6501   nzb 01
```

As another consequence of the constant time requirement, any satellite equipment used with the computer must be tested for operation time. To illustrate, consider the analog to digital converter. There is a time delay between the time an input is called and the time the input is inserted into the computer. During this delay the computer cannot be used for other purposes. This time delay must be found so that it can be taken into consideration in computing total program time. To determine the delay, run a program of the input routine repeated several times and with a stopwatch time the program. Knowing

the total number of analog to digital conversions and the time for these conversions, the time per conversion can be calculated. A similar procedure can be used to determine the digital to analog conversion time.

CHAPTER IV

HYBRID CONTROL POLICY

1. Theory

The process description is

$$\dot{\underline{\theta}} = F\,\underline{\theta}(K) + \underline{D}\,U(K) \qquad \text{IV-1}$$

where $F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $\underline{D} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$,

and in sampled form is

$$\underline{\theta}(K+1) = e^{FT}\,\underline{\theta}(K) + \Delta_n U(n) \qquad \text{IV-2}$$

Let $$\phi = e^{FT} = I + FT + \frac{(FT)^2}{2} + \cdots$$

$$\phi = I + FT + (F)(F)\frac{T^2}{2} + \cdots$$

gives $$\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \qquad \text{IV-3}$$

and $$\Delta_n = \int_0^T e^{F(T-t)}\,D\,dt$$

$$= \int_0^T \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} dt = \int_0^T \begin{bmatrix} t \\ 1 \end{bmatrix} dt = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \qquad \text{IV-4}$$

Substituting into the process equation, we obtain

$$\underline{\theta}(K+1) = \phi\,\underline{X}(K) + \underline{U}(K) \qquad \text{IV-5}$$

It is now desired to get the reference position in 2 samples

$$\underline{\theta}(1) = \phi\,\underline{\theta}(0) + \underline{U}(0)$$

$$\underline{\theta}(2) = \phi\,\underline{\theta}(1) + \underline{U}(1) = 0$$

$$\theta = \phi\,(\phi\,\underline{X}(0) + \underline{U}(0)) + \underline{U}(1)$$

$$\theta = \begin{bmatrix} 1 & 2T \\ 0 & 1 \end{bmatrix}\underline{X}(0) + \begin{bmatrix} 3T^2/2 \\ T \end{bmatrix}\underline{U}(0) + \begin{bmatrix} T^2/2 \\ T \end{bmatrix}\underline{U}(1) \qquad \text{IV-6}$$

Letting $$\underline{U}_n = -a_1\,\theta(n) - a_2\,\dot{\theta}(n), \text{ and} \qquad \text{IV-7}$$

expanding IV-7

$$\theta(0) + 2T\,\dot{\theta}(0) = -\frac{3}{2}T^2U(0) - \frac{T^2}{2}U(1) \qquad \text{IV-6a}$$

$$\dot{\theta}(0) = T\,U(0) + T\,U(1) \quad (\times T/2) \qquad \text{IV-6b}$$

Simultaneous solution of these equations gives

$$U(0) = -\frac{\theta(0)}{T^2} - \frac{3\,\dot{\theta}(0)}{2T}$$

or generally, $U(K) = -\frac{\theta(K)}{T^2} - \frac{3\dot{\theta}(K)}{2T}$, IV-8

the desired control policy. This policy will force the second order system to its reference position in two sampling periods (2T). (The sampling period (T) should not be confused with that alluded to in the state generator discussion in App. II)

2. Memory Required

All lower memory, except 70-77.

Upper memory - 164_8 cells.

3. Subroutines Required

Floating Point Arithmetic Package.

State Generator Program with the "Divide-by-Four" subroutine which it requires.

4. Special Features

The sampling period (T) is adjusted by manually entering the desired counting value in cell 2430 (as the program is herein listed). 1712 for T=1 second, and 730 for T=½ second.

When the sampling period T is changed, the equivalent floating point number must be entered into cells 50 and 51, and its square into cells 52 and 53. These entries give the proper gain value for the chosen sampling period.

5. CDC 160 Computer Control Program

The program to implement the control policy follows.

Summary of Cell Allocations

(including detail of lower memory)

0100 - 1703 floating point arithmetic package
1704 - 1730 Divide-by-4
1731 - 1760 2T/3 computation
1761 - 2303 State generator
2304 - 2414 Control Policy including output

Lower Memory

01-04	floating point arguments	
06	program loop-closing jump (to 1761)	
07	subroutine return jumps	
10-1	UU & L floating point forcing fct	
12-3	TE2U & L $3\ddot{\theta}/2T$	
14-5	TE1U & L θ/T^2	
16-7	2TD3U & L 3/2T	
20	U forcing function	
21-2-3	θA, θB, θC	
24-5	$\theta B-\theta A$, $\theta C-\theta B$, resp.	
26	$(\theta C-\theta B) - (\theta B-\theta A)$	
30-1	$(\theta B-\theta A)$ U & L	
32-3	θBDU & L	
34-5	$(\theta C-\theta B)$ U & L	
36-7	θCDU & L	
40-1	$(\theta C-\theta B) - (\theta B-\theta A)$ U & L	
42-3	θCDDU & L	
44-5	Δ TU & L	
46-7	ΔT2U & L	
50-1	TU & L	
52-3	T2U & L	
54-5	2U & L	
56-7	3U & L	
61	Divided by 4 (1704)	
62	FSB	0317
63	FAD	0335
64	FMU	1315
65	FDV	1533
66	FIX-FLO	0100
67	FLO-FIX	0162

Unused

00,05,27 & 60

FORM 2T/3

```
1731  2054  ldd 54
1732  4001  std 01
1733  2055  ldd 55
1734  4002  std 02
1735  2050  ldd 50
1736  4003  std 03
1737  2051  ldd 51
1740  4004  std 04

1741  0101  pta
1742  0604  adn 04
1743  4007  std 07
1744  7064  jpi 64
1745  2056  ldd 56
1746  4003  std 03
1747  2057  ldd 57
1750  4004  std 04

1751  0101  pta
1752  0604  adn 04
1753  4007  std 07
1754  7065  jpi 65
1755  2001  ldd 01
1756  4016  std 16
1757  2002  ldd 02
1760  4017  std 17
```

Continue into the main loop of the State Generator.

```
2304  0101  pta
2305  0605  adn 05
2306  4007  std 07
2307  2023  ldd 23

2310  7066  jpi 66      θC to floating point
2311  2052  ldd 52
2312  4003  std 03
2313  2053  ldd 53
2314  4004  std 04
2315  0101  pta
2316  0604  adn 04
2317  4007  std 07

2320  7065  jpi 65      θ(K)/T²
2321  2001  ldd 01
2322  4014  std 14
2323  2002  ldd 02
2324  4015  std 15
2325  2036  ldd 36
2326  4001  std 01
2327  2037  ldd 37

2330  4002  std 02
2331  2016  ldd 16
2332  4003  std 03
2333  2017  ldd 17
2334  4004  std 04
2335  0101  pta
2336  0604  adn 04
2337  4007  std 07

2340  7065  jpi 65      θ DOT/(2T/3)
2341  2001  ldd 01
2342  4012  std 12
2343  2002  ldd 02
2344  4013  std 13
2345  2014  ldd 14
2346  4003  std 03
2347  2015  ldd 15
```

```
2350  4004  std 04
2351  0101  pta
2352  0604  adn 04
2353  4007  std 07
2354  7063  jpi 63     add to form -U
2355  2001  ldd 01
2356  4010  std 10
2357  2002  ldd 02

2360  4011  std 11
2361  0101  pta
2362  0604  adn 04
2363  4007  std 07
2364  7067  jpi 67     -U  to fixed point
2365  1600  scc
2366  7777  7777       complement gives +U
2367  4020  std 20

2370  7500  exf 00
2371  2401  2401
2372  7304  out 04     output  U
2373  0021  0021
2374  6103  nzf 03
2375  6002  zjf 02
2376  0020  0020
2377  2200  ldc 00

2400  1712  1712
2401  4211  stf 11
2402  2200  ldf 00
2403  0117  0117
2404  0701  sbn 01
2405  6501  nzb 01
2406  2204  ldf 04
2407  0701  sbn 01

2410  6507  nzb 07
2411  6202  pjf 02
2412  0000  0000
2413  7006  jpi 06     jump to close main loop, (0006)=1761
2414  7717  hlt 17
```

CHAPTER V

PROCEDURE

The programs for generating the state variable (App. II) and control function (Chap. IV) were tested using the circuits shown in Figs. 2 and 1, respectively.

The TR-10 biasing levels, Fig. 2, for closed loop control were very sensitive. A small error yielded completely erroneous results. This was not a problem in the state variable test. The limiter was provided to protect the A/D converter from inputs beyond its range (0 to -10 volts). The linearity of the A/D and D/A was determined to be satisfactory before operating the system. Initial conditions were chosen to avoid saturating the voltage limits (e.g. $U(0) = \theta_0/T^2$); also, saturation of the float-to-fixed point program will occur whenever the floating point number exceeds the largest possible positive and negative fixed point numbers, 3777 and 4000.

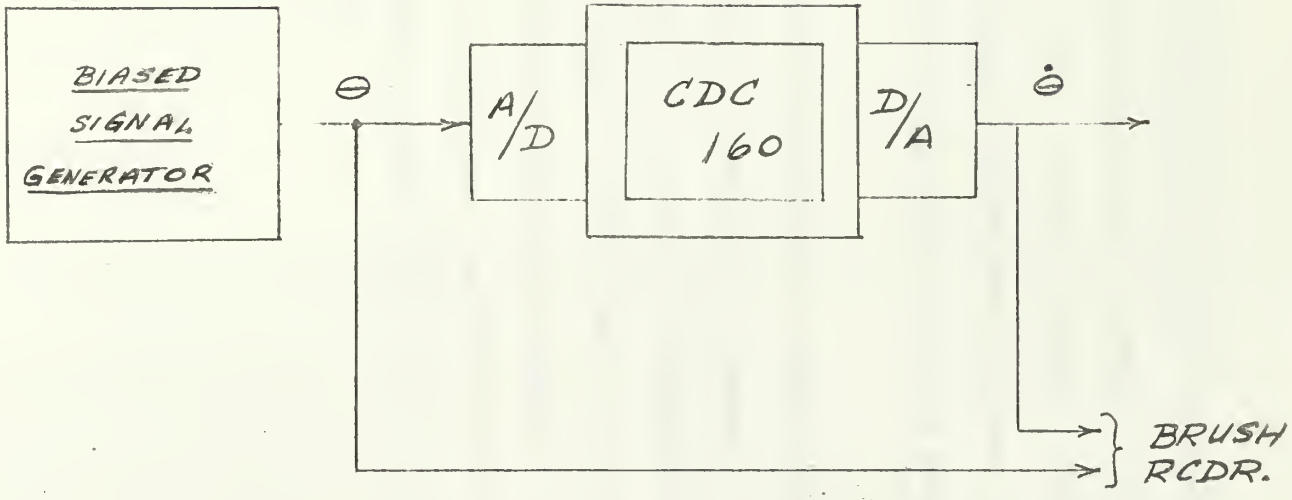


FIGURE 2

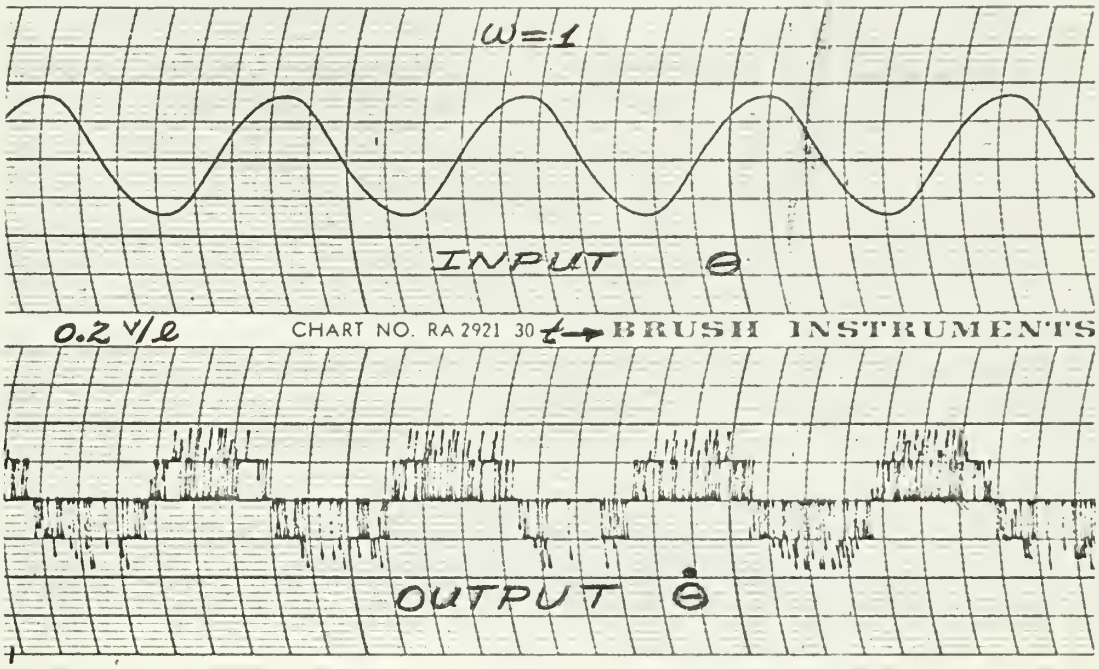
STATE GENERATOR TEST SYSTEM

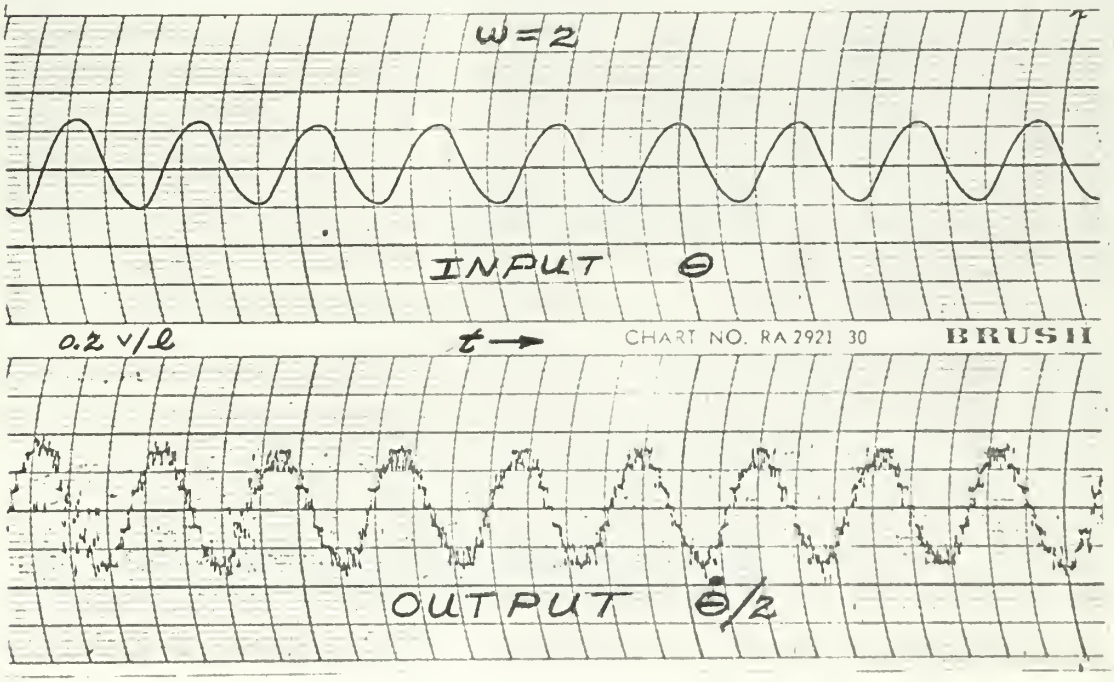
CHAPTER VI

RESULTS AND DISCUSSION

The program for generating the first derivative is shown in App. II. Using this program and the system of Fig. 2, the average peak-to-peak noise level is about 2.0 volts. Except for the noise effects, the derivatives obtained were reliable. Any error in amplitude and/or phase being within the noise envelope as shown in Fig. 3.

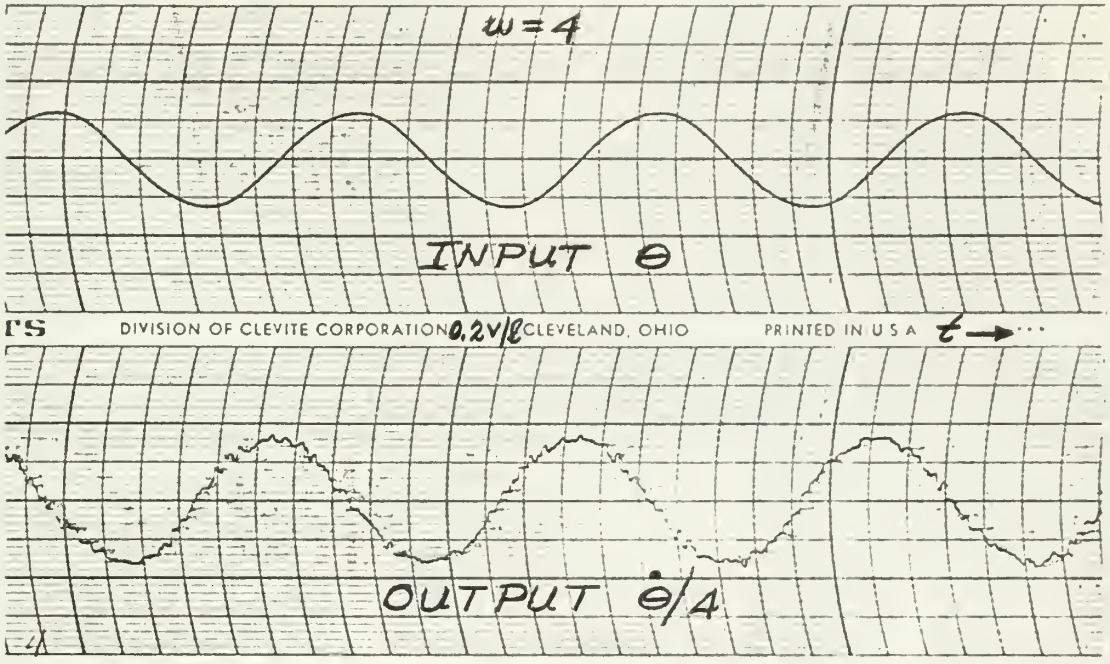
The program for implementing the control policy is shown in Chap. IV. The results of testing this program with an imbedded step are shown in Fig. 4. If noise is relatively low at the instants of sampling, the control is nearly ideal, as in Fig. 4a. However, when noise is appreciable the result obtained is like that shown in Figs. 4b and 4c. Fig. 4b shows the effect of noise making the $\ddot{\theta}$ term of the control policy too large, while Fig. 4c shows the effect when $\dot{\theta}$ is made too small.

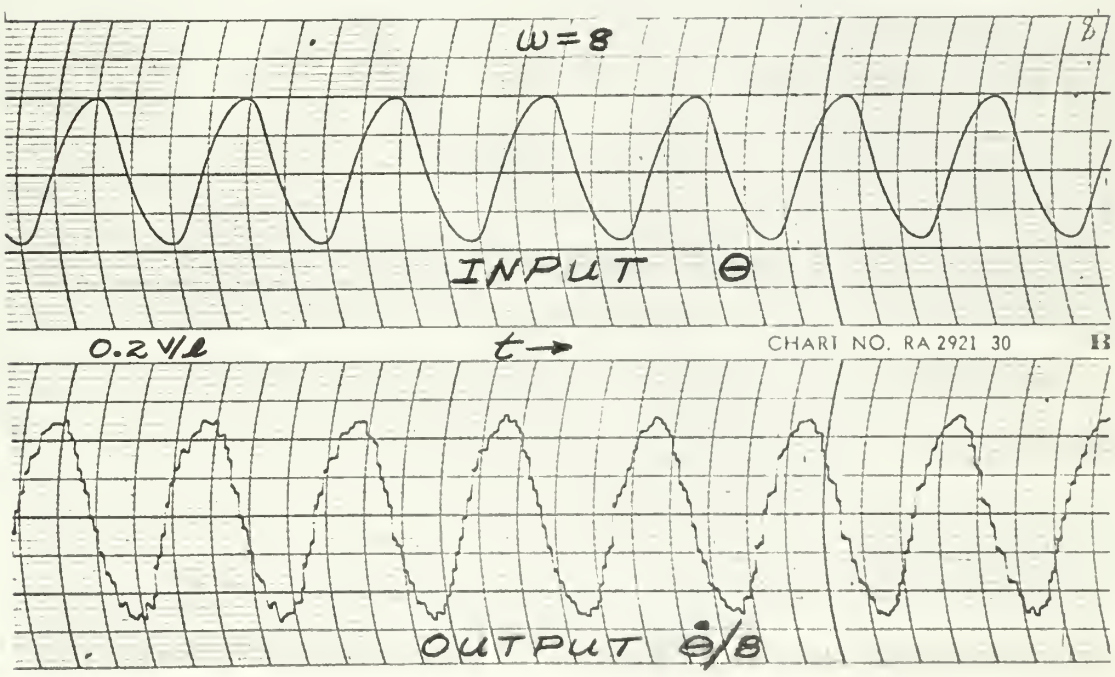




DERIVATIVE RESULTS

FIGURE 3





DERIVATIVE RESULTS (con't)

FIGURE 3

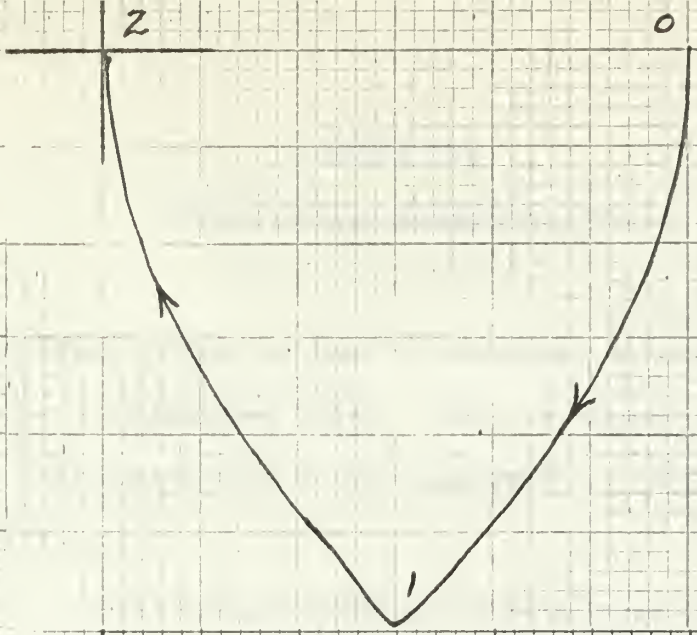


Fig. 4a
IDEAL

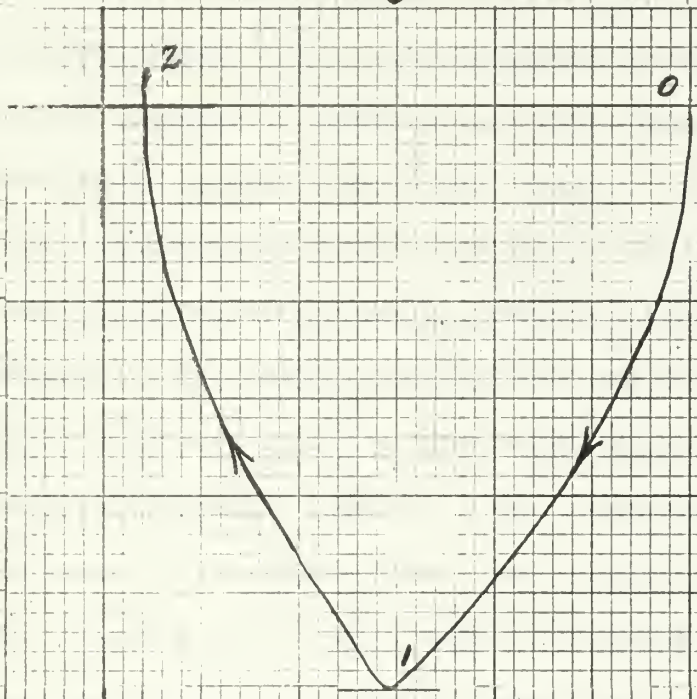


Fig. 4b
$\dot{\theta}$ TOO LARGE

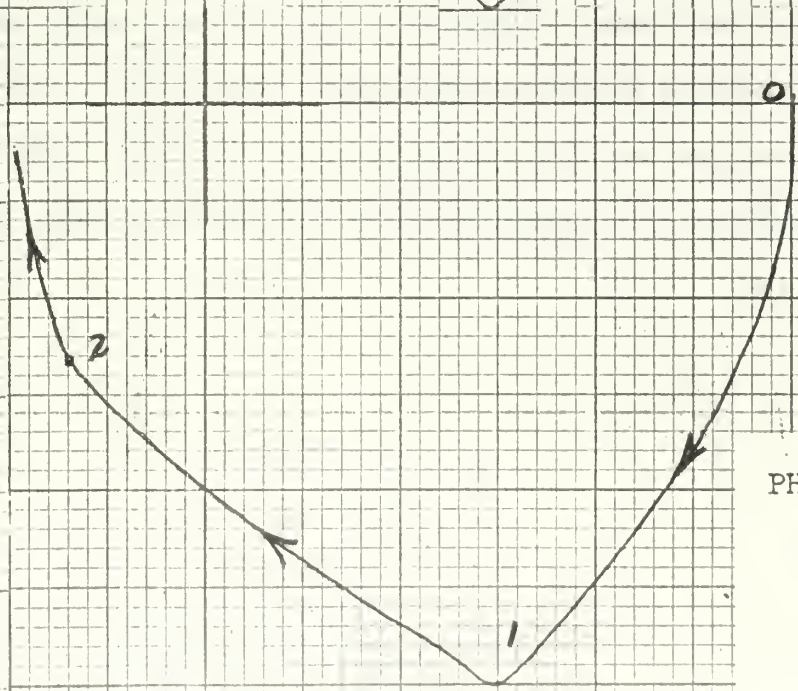


Fig. 4c
$\dot{\theta}$ TOO SMALL

PHASE TRAJECTORIES OF
THE HYBRID SYSTEM
Figure 4

CHAPTER VII

CONCLUSION AND RECOMMENDATIONS

The control policy has been demonstrated to be effective when noise is not a significant factor. This is seldom the situation. The policy as implemented here is very sensitive to noise of the level of several millivolts.

It was further demonstrated that a good first derivative could be digitally determined, but is very sensitive to input noise.

It was not possible to determine the major source(s) of noise, but the test signal generator was highly suspect. The generator probably has a high frequency ripple of several millivolts in its output, thus giving the noisy results previously discussed.

Improvement of the results obtained here can be obtained either by eliminating noise or turning to more sophisticated programming, such as predictor-corrector schemes. Noise elimination is all but impractical except in the laboratory. More sophisticated programs will result in much higher "costs" than the programs presented here. These costs will be mainly in computor usage-time and memory space required. Also, while the program utilized here was almost instantaneous, some of the more sophisticated ones may involve significant time (phase) delays for which adjustment will be necessary.

Further investigations along the approach used here should initially determine the noise characteristics of all equipments considered for use in order that excessively noisy signals are not introduced.

BIBLIOGRAPHY

1. Control Data Corporation, CDC 160 Computer Programming Manual.

2. Electronic Associates, Inc., PACE TR-10 Transistorized Analog Computer Operator's Handbook.

3. Lecture Notes, Course EE-414A, H. A. Titus, U. S. Naval Postgraduate School, Monterey, California, September 1963.

APPENDIX I

CDC 160 CONSTANT TIME FLOATING POINT

ARITHMETIC PACKAGE

The Constant Time Floating Point Package contains routines for fixed point to floating point and floating point to fixed point conversions and floating point add, subtract, multiply and divide. These subroutines have been programmed so that the same amount of time is required for each operation regardless of the magnitude of the input quantity. The following paragraphs contain some general information and notes on the package and its use.

1. Floating Point Number Format

The floating point number consists of eight octal digits and requires two storage cells. One cell (lower) contains the significant figures of the number. To form a positive floating point number the fixed point octal number is written in binary form and the binary point shifted to the left of the first significant (one) bit. The octal number of shifts required added to 2000_8 becomes the upper portion of the floating point number. The lower portion is the significant bits in octal form, with the binary point understood to be at the left. To form a negative floating point number the same procedure is followed except the upper and lower portions of the number are "sevens complemented".

Example:

$$3721._8 = 011\ 111\ 010\ 001._2$$

$$3721._8 = .111\ 110\ 100\ 010_2 \times 2^{13}_8$$

$$= .7642_8 \times 2^{13}_8$$

The positive floating point number is $\begin{matrix}2013\\7642\end{matrix}$ and the negative is $\begin{matrix}5764\\0135\end{matrix}$.

2. Significant Figures

This package recognizes the following configurations as zero:

Upper	0000	0000	XXXX
Lower	0000	XXXX	0000

In the interest of conserving time the last significant figure has not been rounded off in any of the routines in this package. In the floating-to-fixed point conversions, if the magnitude of the number exceeds capacity of the computer in fixed point, the output will be 3777_8 if the number is positive or 4000_8 if the number is negative.

3. Cells Required

The package requires the following cells:

01	X upper stowage
02	X lower stowage
03	Y upper stowage
04	Y lower stowage
07	XXXX return jump address
0100 to 1703	floating point package

4. Jump Routines

The following routines are recommended for jumping into the floating point package.

Fixed point to floating point conversion:

```
0101  pts
0605 adn 05
4007 std 07
20XX ldd XX
70YY jpi YY
```

XX is the number of the cell in which the fixed point number is stored.

YY is the number of the cell in which the address of the fixed-to-floating point routine is stored.

All other operations:

```
0101  pta      XX is the number of the cell in which the
0604  adn 04       jump address of the operation is stored.
4007  std 07
70XX  jpi XX
```

In each case control is returned to the main program at the address following the jpi instruction.

5. Summary

OPERATION (CODE)	JUMP ADDRESS	DESCRIPTION	ANSWER IN	TIME CYCLES
Fixed to floating point conversion. (FX-FL)	0100	(A) → X	01 02	167
Floating to fixed point conversion. (FL-FX)	0162	X → A	A	216
Floating point subtraction. (FSB)	0317	X - Y	01 02	319
Floating point addition. (FAD)	0323	X + Y	01 02	303
Floating point multiplication (FMU)	1315	X * Y	01 02	674
Floating point division. (FDV)	1533	X / Y	01 02	380

FLOATING POINT ARITHMETIC PACKAGE

```
0100  4002  std 02     FX-FL.  Store fixed point number in cell 02.
0101  0513  lcn 13     Set count 1 with 13.
0102  4255  stf 55
0103  0400  ldn 00     Zeroize count 2 and cell 01.
0104  4254  stf 54
0105  4001  std 01
0106  2002  ldd 02     Store fixed point number for later sign check.
0107  4247  stf 47

0110  6204  pjf 04
0111  2402  lcd 02     Number is negative, complement and restore.
0112  4002  std 02
0113  6204  pjf 04
0114  0401  ldn 01     Add one to count 2 to equalize positive and
0115  4243  stf 43        negative paths.
0116  2002  ldd 02
0117  6033  zjf 33     If number is zero jump forward.

0120  5401  aod 01     Shift cell 02 until first bit is a one bit.
0121  5636  aof 36         For each shift add one to cell 01 and
0122  4402  srd 02         count 1.  Cell 02 now contains the lower
0123  6603  pjb 03         part of the floating point number.
0124  2200  ldc 00     Subtract cell 01 from 2014 to form the upper
0125  2014  2014             part of the floating point number and
0126  3401  sbd 01         restore.
0127  4001  std 01

0130  2226  ldf 26     Check sign of original number.
0131  6205  pjf 05
0132  2401  lcd 01     Number is negative, complement upper and
0133  4001  std 01         lower parts and restore.
0134  2402  lcd 02
0135  4002  std 02
0136  2222  ldf 22     Form count.  Count = number of delay loops
0137  3620  sbf 20                   = Count 2 - Count 1

0140  4221  stf 21
0141  6010  zjf 10     No delay loops are necessary.
0142  2217  ldf 17     Delay loops to equalize time.
0143  0701  sbn 01
0144  4215  stf 15
0145  0300  nop
0146  0300  nop
0147  0300  nop
```

```
0150  6506  nzb 06
0151  7007  jpi 07     Return jump to main program.
0152  0415  ldn 15     Form count for fixed point number
0153  0300  nop            equal to zero.
0154  0300  nop
0155  6615  pjb 15
0156  0000             Sign check stowage.
0157  0000             Count 1 stowage. Count 1 = 13 - No. of shifts.

0160  0000             Count 2 stowage.
0161  0000             Count stowage.
0162  2001  ldd 01     FL-FX.  Load upper part of floating point
0163  4260  stf 60         number and store as sign check.
0164  6305  njf 05
0165  0404  ldn 04     Number is positive. Time delay to equalize
0166  0701  sbn 01         positive and negative paths.
0167  6501  nzb 01

0170  6005  zjf 05
0171  2401  lcd 01     Number is negative. Complement upper and
0172  4001  std 01         lower parts and restore.
0173  2402  lcd 02
0174  4002  std 02
0175  2002  ldd 02
0176  6051  zjf 51     If number is zero jump to 0247.
0177  2001  ldd 01

0200  3600  sbc 00     Subtract 2000to find exponent. If positive
0201  2000  2000           store on shift.
0202  6046  zjf 46     Exponent is zero. Jump to 0250.
0203  6346  njf 46     Exponent is negative. Jump to 0251.
0204  4240  stf 40
0205  0714  sbn 14     Subtract 14 from shift and stow result
0206  4237  stf 37         in count. Count = No. delay loops.
0207  6202  pjf 02     If count is positive,number will exceed

0210  6302  njf 02         capacity. Jump to 0252.
0211  6141  nzf 41
0212  0400  ldn 00
0213  4233  stf 33     Zeroize mask and form complement of shift.
0214  2630  lcf 30
0215  4227  stf 27
0216  4402  srd 02     Shift cell 02 shift number of times. At
0217  4627  srf 27         same time form mask.

0220  5626  aof 26
0221  5623  aof 23
0222  6504  nzb 04
0223  2222  ldf 22     Delay loops to equalize time.
0224  6006  zjf 06
0225  0404  ldn 04
0226  0701  sbn 01
0227  6501  nzb 01
```

```
0230  5615  aof 15
0231  6504  nzb 04
0232  2002  ldd 02      Mask cell 02 and restore. Cell 02 contains
0233  1213  lpf 13          the fixed point number.
0234  4002  std 02
0235  2206  ldf 06      Check sign of original number. If negative
0236  6303  njf 03          complement and load in A register. If
0237  2002  ldd 02          positive load in A register. Re-

0240  7007  jpi 07          turn jump to main program.
0241  2402  lcd 02
0242  7007  jpi 07
0243  0000              Check stowage.
0244  0000              Shift stowage.
0245  0000              Count stowage.
0246  0000              Mask stowage.
0247  6004  zjf 04      JUmp to0253.

0250  6012  zjf 12      Jump to 0262.
0251  6342  njf 24      Jump tp 0275.
0252  6132  nzf 32      Jump to 0304.
0253  2200  ldc 00      Floating point number is zero. Load zero
0254  0137  0137            in A register, delay and return jump
0255  0701  sbn 01          to main program.
0256  6501  nzb 01
0257  0300  nop

0260  0400  ldn 00
0261  7007  jpi 07
0262  2200  ldc 00      Floating point number is larger than 0.5 and
0263  0133  0133            less than 1. Load 0001 in A register,
0264  0701  sbn 01          delay, and return jump to main program.
0265  6501  nzb 01
0266  0300  nop
0267  2324  ldb 24

0270  6303  njf 03
0271  0401  ldn 01
0272  7007  jpi 07
0273  0501  lcn 01
0274  7007  jpi 07
0275  2200  ldc 00      Floating point number is greater than zero
0276  0134  0134            but less than 0.5. Load zero in A re-
0277  0701  sbn 01          gister, delay and return jump to main

0300  6501  nzb 01          program.
0301  0300  nop
0302  0400  ldn 00
0303  7007  jpi 07
0304  2200  ldc 00      Floating point number exceeds capacity of
0305  0126  0126            computer. If number is positive load
0306  0701  sbn 01          3777 in A register, if negative load
0307  6501  nzb 01          4000, delay, and return jump to main
```

```
0310  2345  ldb 45        program.
0311  6304  njf 04
0312  2200 lldc 00
0313  3777  3777
0314  7007  jpi 07
0315  2702  lcb 02
0316  7007  jpi 07
0317  2003  ldd 03     FSB.  Zero check Y upper and lower. If

0320  6104  nzf 04        either are zero jump to FAD. If both
0321  4003  std 03        are non-zero, complement, restore and
0322  0300  nop           jump to FAD.
0323  6003  zjf 03
0324  2403  lcd 03
0325  4003  std 03
0326  2004  ldd 04
0327  6104  nzf 04

0330  4004  std 04
0331  0300  nop
0332  6003  zjf 03
0333  2404  lcd 04
0334  4004  std 04
0335  2001  ldd 01     FAD.  Sign and zero check X.
0336  6017  zjf 17
0337  6353  njf 53

0340  2002  ldd 02
0341  6016  zjf 16
0342  2003  ldd 03     X is positive and non- zero. Sign and
0343  6042  zjf 42        zero check Y.
0344  6305  njf 05
0345  2004  ldd 04
0346  6041  zjf 41
0347  7101  jfi 01     X and Y are positive and non-zero. Jump

0350  1077  1077           to 1077.
0351  2004  ldd 04
0352  6035  zjf 35
0353  7101  jfi 01     X is positive and Y is negative. Both
0354  0653  0653           are non-zero. Jump to 0653.
0355  2001  ldd 01     Time delay.  NOTE: X is zero for 0355
0356  2001  ldd 01                        to 0404.
0357  2003  ldd 03     Zero check Y upper.

0360  6105  nzf 05
0361  0402  ldn 02     Time delay.
0362  0701  sbn 01
0363  6501  nzb 01
0364  6010  zjf 10
0365  4001  std 01     Y upper non-zero. Store in cell 01.
0366  2004  ldd 04     Zero check Y lower.
0367  6006  zjf 06
```

```
0370  4002  std 02    Y lower is non-zero. Store in cell 02.
0371  2200  ldc 00    Answer is now in cells 01 and 02. Delay
0372  0213  0213            and return jump to main program.
0373  0300  nop
0374  6106  nzf 06
0375  0400  ldn 00    Answer is zero. Load zero in cells 01 and
0376  4001  std 01          02, delay and return jump to main
0377  4002  std 02          program.

0400  2200  ldc 00
0401  0212  0212
0402  0701  sbn 01
0403  6501  nzb 01
0404  7007  jpi 07
0405  2003  ldd 03    Y is zero. Delay and return jump to main
0406  2003  ldd 03          program.
0407  2200  ldc 00

0410  0216  0210
0411  6507  nzb 07
0412  2002  ldd 02    Zero check X lower. NOTE: X is negative
0413  6434  zjb 34                              for 0412 to 0424.
0414  2003  ldd 03    Zero and sign check Y upper.
0415  6410  zjb 10
0416  6205  pjf 05
0417  2004  ldd 04    Y is negative. Zero check Y lower.

0420  6411  zjb 11
0421  7101  jpi 01    X and Y are negative. Jump to 1105.
0422  1105  1105
0423  2004  ldd 04    Y is positive. Zero check Y lower.
0424  6415  zjb 15
0425  2401  lcd 01    Subtract upper parts and store result
0426  3403  sbd 03          in count 1. NOTE: 0425 to 0652 is
0427  4275  stf 75          X negative and Y positive routine.

0430  6077  zjf 77    Exponents equal. Jump to 0527.
0431  6277  pjf 77    X exponent larger. Jump to 0530.
0432  0614  adn 14    Add 14 to count 1 to form shift.
0433  6056  zjf 56    Y is much larger than X. Jump to 0511.
0434  6356  njf 56
0435  1600  scc 00    Complement and restore shift.
0436  7777  7777
0437  4266  stf 66

0440  0400  ldn 00    Zeroize mask.
0441  4265  stf 65
0442  4402  srd 02    Shift X lower to equalize exponents.
0443  4663  srf 63          Form mask. No. of shifts = shift.
0444  5662  aof 62
0445  5660  aof 60
0446  6504  nzb 04
0447  2402  lcd 02    Mask and restore X lower.
```

```
0450  1256  lpf 56
0451  4002  std 02
0452  0513  lcn 13     Load -13 in count 2.
0453  4253  stf 53
0454  2004  ldd 04     Subtract X and Y lower and store result
0455  3402  sbd 02         on cell 02.
0456  4002  std 02
0457  6206  pjf 06

0460  2003  ldd 03     First bit is a one bit. No shift is
0461  4001  std 01         necessary. Store exponent in cell
0462  0401  ldn 01         01. Time delay.
0463  0701  sbn 01
0464  6010  zjf 10
0465  5641  aof 41     Shift cell 02 until first bit is a one
0466  4402  srd 02         bit. Modify count 2 for each shift.
0467  6602  pjb 02

0470  2636  lcf 36     Form exponent and store in cell 01.
0471  0713  sbn 13
0472  3003  add 03
0473  4001  std 01
0474  2230  ldf 30     Zero check count 1.
0475  6006  zjf 06
0476  0404  ldn 04     Count 1 delay loops.
0477  0701  sbn 01

0500  6501  nzb 01
0501  5623  aof 23
0502  6504  nzb 04
0503  2223  ldf 23     Zero check count 2.
0504  6004  zjf 04
0505  4220  stf 20     Count 2 dealy loops.
0506  5620  aof 20
0507  6502  nzb 02

0510  7007  jpi 07     Return jump to main program.
0511  0300  nop        Y is much greater than X. Load Y in
0512  2003  ldd 03         cells 01 and 02, dealy and re-
0513  4001  std 01         turn jump to main program.
0514  2004  ldd 04
0515  4002  std 02
0516  2200  ldc 00
0517  0204  0204

0520  0300  nop
0521  0701  sbn 01
0522  6501  nzb 01
0523  7007  jpi 07
0524  0000              Count 1 stowage.
0525  0000              Shift stowage.
0526  0000              Mask and count 2 stowage.
0527  6055  zjf 55     Exponents equal. Jump to 0604.
```

```
0530  0714  sbn 14   X exponent larger. Form shift.
0531  4304  stb 04
0532  6244  pjf 44   X is much larger than Y. Jump to 0576.
0533  0400  ldn 00   Zeroize mask.
0534  4306  stb 06
0535  4404  srd 04   Shift Y lower to equalize exponents.
0536  4710  srb 10       Form mask. No. of shifts = shift.
0537  5711  aob 11

0540  5713  aob 13
0541  6504  nzb 04
0542  2004  ldd 04   Mask and restore Y lower.
0543  1315  lpb 15
0544  4004  std 04
0545  0513  lcn 13   Load -13 in count 2.
0546  4320  stb 20
0547  2402  lcd 02   Subtract X and Y lower and store re-

0550  3404  sbd 04       sult in cell 02.
0551  4002  std 02
0552  6205  pjf 05
0553  0402  ldn 02   Time delay.
0554  0701  sbn 01
0555  6501  nzb 01
0556  6007  zjf 07
0557  5731  aob 31   Shift cell 02 until first bit is a

0560  4402  srd 02       one bit. Modify count 2 for each
0561  6602  pjb 02       shift.
0562  2334  ldb 34   Form exponent and store in cell 01.
0563  0613  adn 13
0564  5001  rad 01
0565  2402  lcd 02   Answer is negative. Complement and
0566  4002  std 02       restore cell 02.
0567  2743  lcb 43   Modify count 1 to equalize delay.

0570  0601  adn 01
0571  4345  stb 45
0572  2200  ldc 00   Modify program for proper jump.
0573  6531  6531
0574  4347  stb 47   Jump back to counts 1 and 2 dealy loops.
0575  6550  nzb 50
0576  2200  ldc 00   X much larger than Y delay.
0577  0207  0207

0600  0300  nop
0601  0701  sbn 01
0602  6501  nzb 01
0603  7007  jpi 07
0604  0515  lcn 15   X and Y exponents equal. Form count 3.
0605  4245  stf 45
0606  2402  lcd 02   Subtract X and Y lower and store in
0607  3404  sbd 04       cell 02.
```

```
0610  4002  std 02
0611  6105  nzf 05
0612  4001  std 01    X and Y are equal. Answer is zero. Jump
0613  2200  ldc 00          to delay routine.
0614  0201  0201
0615  6134  nzf 34
0616  6314  njf 14
0617  5633  aof 33    X is greater than Y. Left shift cell 02

0620  4402  srd 02        until first bit is a one bit. Modify
0621  6602  pjb 02        count 3 for each shift.
0622  2402  lcd 02
0623  4002  std 02
0624  2226  ldf 26    Form exponent and store in cell 01.
0625  0615  adn 15
0626  5001  rad 01
0627  0300  nop       Time delay.

0630  0400  ldn 00
0631  6013  zjf 13
0632  2402  lcd 02    X is less than Y. Left shift cell 02
0633  4002  std 02        until first bit is a one bit.
0634  5616  aof 16        Modify count 3 for each shift.
0635  4402  srd 02
0636  6602  pjb 02
0637  2613  lcf 13    Form exponent and store in cell 01.

0640  0715  sbn 15
0641  3003  add 03
0642  4001  std 01
0643  0300  nop       Count 3 delay loops.
0644  5606  aof 06
0645  4205  stf 05
0646  6502  nzb 02
0647  2200  ldc 00    Time delay.

0650  0117  0117
0651  6551  nzb 51
0652  0000            Count 3 stowage.
0653  2403  lcd 03    Subtract upper parts and store result in
0654  3401  sbd 01       count 1.  NOTE: X is positive and Y is
0655  4267  stf 67       for 0653 to 1076.
0656  6071  zjf 71    Exponents equal. Jump to 0747.
0657  6271  pjf 71    Y larger than X. Jump to 0750.

0660  0614  adn 14    Add 14 to count 1 to form shift.
0661  6054  zjf 54    X is much larger than Y. Jump to 0735.
0662  6354  njf 54
0663  1600  scc 00    Complement and restore shift.
0664  7777  7777
0665  4260  stf 60
0666  0400  ldn 00    Zeroize mask.
0667  4257  stf 57
```

```
0670  4404  srd 04   Shift Y lower to equalize exponents.
0671  4655  srf 55       Form mask. No. shifts = Shift.
0672  5654  aof 54
0673  5652  aof 52
0674  6504  nzb 04
0675  2404  lcd 04   Mask and restore Y lower.
0676  1250  lpf 50
0677  4004  std 04

0700  0513  lcn 13   Load -13 in count 2.
0701  4245  stf 45
0702  2002  ldd 02   Subtract lower parts and store in
0703  3404  sbd 04       cell 02.
0704  4002  std 02
0705  6205  pjf 05
0706  0402  ldn 02   First bit is a one bit. No shift is
0707  0701  sbn 01       needed. Time delay.

0710  6501  nzb 01
0711  6007  zjf 07
0712  5634  aof 34   Left shift cell 02 until first bit is
0713  4402  srd 02       a one bit. Modify count 2 for each
0714  6602  pjb 02       shift.
0715  2631  lcf 31   Form exponent and store in cell 01.
0716  0713  sbn 13
0717  5001  rad 01

0720  2224  ldf 24   Zero check count 1.
0721  6006  zjf 06
0722  0404  ldn 04   Count 1 delay loops.
0723  0701  sbn 01
0724  6501  nzb 01
0725  5617  aof 17
0726  6504  nzb 04
0727  2217  ldf 17   Zero check count 2.

0730  6004  zjf 04
0731  4214  stf 14   Count 2 delay loops.
0732  5614  aof 14
0733  6502  nzb 02
0734  7007  jpi 07   Return jump to main program.
0735  0300  nop      X is much larger than Y. Delay and
0736  2200  ldc 00       return jump to main program.
0737  0207  0207

0740  0300  nop
0741  0701  sbn 01
0742  6501  nzb 01
0743  7007  jpi 07
0744  0000            Count 1 storage.
0745  0000            Shift storage.
0746  0000            Mask and count 2 storage.
0747  6062  zjf 62    Exponents equal. Jump to 1017.
```

```
0750  0714  sbn 14
0751  4304  stb 04
0752  6245  pjf 45    Y much larger than X. Jump to 1017.
0753  0400  ldn 00    Zeroize mask.
0754  4306  stb 06
0755  4402  srd 02    Shift X lower to equalize exponents.
0756  4710  srb 10        Form mask. No. shifts = Shift.
0757  5711  aob 11

0760  5713  aob 13
0761  6504  nzb 04
0762  2002  ldd 02    Mask and restore X lower.
0763  1315  lpb 15
0764  4002  std 02
0765  0513  lcn 13    Load -13 in count 2.
0766  4320  stb 20
0767  2404  lcd 04    Subtract lower parts and store result

0770  3402  sbd 02        in cell 02.
0771  4002  std 02
0772  6205  pjf 05    First bit is a one bit. No shift is
0773  2003  ldd 03        necessary. Store exponent in cell
0774  4001  std 01        01.
0775  2003  ldd 03    Time delay.
0776  6110  nzf 10
0777  5731  aob 31    Left shift cell 02 until first bit is

1000  4402  srd 02        a one bit. Modify count 2 for each
1001  6602  pjb 02        shift.
1002  2334  ldb 34    Form exponent and store in cell 01.
1003  0613  adn 13
1004  3003  add 03
1005  4001  std 01
1006  2402  lcd 02    Complement and restore cell 02.
1007  4002  std 02

1010  2744  lcb 44    Modify count 1 to equalize time delay.
1011  0601  adn 01
1012  4346  stb 46
1013  0300  nop       Jump to count 1 and 2 delay loops.
1014  6002  zjf 02
1015  6575  nzb 75
1016  6476  zjb 76
1017  2003  ldd 03    Y is much larger than X. Store Y in

1020  4001  std 01        cells 01 and 02, delay and return
1021  2004  ldd 04        jump to main program.
1022  4002  std 02
1023  2200  ldc 00
1024  0201  0201
1025  0300  nop
1026  0701  sbn 01
1027  6501  nzb 01
```

```
1030  7007  jpi 07
1031  0515  lcn 15   Load -15 in count 3.
1032  4244  stf 44
1033  2404  lcd 04   Subtract lower parts and store result
1034  3402  sbd 02       in cell 02.
1035  4002  std 02
1036  6105  nzf 05
1037  4001  std 01   X and Y are equal. Answer is zero. Jump

1040  2200  ldc 00       to delay routine.
1041  0201  0201
1042  6514  nzb 14
1043  6213  pjf 13   X is larger than Y. Shift complement
1044  2402  lcd 02       02 until first bit is a one bit.
1045  4002  std 02       Modify count 3 for each shift.
1046  5630  aof 30
1047  4402  srd 02

1050  6602  pjb 02
1051  2625  lcf 25   Form exponent and store in cell 01.
1052  0715  sbn 15
1053  5001  rad 01
1054  0300  nop
1055  6212  pjf 12
1056  5620  aof 20   Y is larger than X. Shift cell 02
1057  4402  srd 02       until first bit is a one bit.

1060  6602  pjb 02       Modify count 3 for each shift.
1061  2402  lcd 02   Complement and restore cell 02.
1062  4002  std 02
1063  2213  ldf 13   Form exponent and store in cell 01.
1064  0615  adn 15
1065  3003  add 03
1066  4001  std 01
1067  2207  ldf 07   Count 3 delay loops.

1070  0601  adn 01
1071  4205  stf 05
1072  6503  nzb 03
1073  2200  ldc 00   Time delay.
1074  0116  0116
1075  6547  nzb 47
1076  0000           Count 3 stowage.
1077  0411  ldn 11   X and Y are positive. Delay and store

1100  0701  sbn 01       sign check.
1101  6501  nzb 01
1102  0400  ldn 00
1103  4275  stf 75
1104  6013  zjf 13
1105  2401  lcd 01   X and Y are negative. Complement and
1106  4001  std 01       restore. Store sign check.
1107  2402  lcd 02
```

```
1110  4002  std 02
1111  2403  lcd 03
1112  4003  std 03
1113  2404  lcd 04
1114  4004  std 04
1115  0401  ldn 01
1116  4262  stf 62
1117  2003  ldd 03   Subtract exponents and store result in

1120  3401  sbd 01       count.
1121  4260  stf 60
1122  6064  zjf 64   Exponents are equal. Jump to 1206.
1123  6264  pjf 64   Y is larger than X. Jump to 1207.
1124  0614  adn 14   Add 14 to count to form shift.
1125  6057  zjf 57   X is much larger than Y. Jump to 1204.
1126  6357  njf 57
1127  1600  scc 00   Complement and restore shift.

1130  7777  7777
1131  4251  stf 51
1132  0400  ldn 00   Zeroize mask.
1133  4250  stf 50
1134  4404  srd 04   Shift Y lower to equalize exponents.
1135  4646  srf 46       Form mask. No. of shifts = Shift.
1136  5645  aof 45
1137  5643  aof 43

1140  6504  nzb 04
1141  2004  ldd 04   Mask Y lower and restore.
1142  1241  lpf 41
1143  4004  std 04
1144  2002  ldd 02   Mask first bit of X lower and add Y
1145  1200  lpc 00       lower
1146  3777  3777
1147  3004  add 04

1150  6310  njf 10   No overflow. Replace first one bit.
1151  1600  scc 00
1152  40000 4000
1153  4002  std 02
1154  0403  ldn 03   Time delay.
1155  0701  sbn 01
1156  6501  nzb 01
1157  6011  zjf 11

1160  0111  ls 6     There is overflow. Right shift one
1161  0110  ls 3         place and replace first one bit.
1162  0103  ls 2         Store in cell 02.
1163  1315  lpb 15
1164  3200  adc 00
1165  2000  2000
1166  4002  std 02
1167  5401  aod 01   Modify exponent.
```

```
1170  2211  ldf 11    Zero check count.
1171  6006  zjf 06
1172  0404  ldn 04    Count delay loops.
1173  0701  sbn 01
1174  6501  nzb 01
1175  5604  aof 04
1176  6504  nzb 04
1177  6044  zjf 44    Jump to delay routine.

1200  0000            Check stowage.
1201  0000            Count stowage.
1202  0000            Shift stowage.
1203  0000            Mask stowage.
1204  0501  lcn 01    X is much larger than Y. Jump to 1262.
1205  6355  njf 55
1206  6066  zjf 66    Exponents equal. Jump to 1274.
1207  0714  sbn 14    Y is larger than X. Form shift.

1210  4306  stb 06
1211  6254  pjf 54    Y is much larger than X. Jump to 1265.
1212  0400  ldn 00    Zeroize mask.
1213  4310  stb 10
1214  4402  srd 02    Shift X lower to equalize exponents.
1215  4712  srb 12        Form mask. No. of shifts = Shift.
1216  5713  aob 13
1217  5715  aob 15

1220  6504  nzb 04
1221  2002  ldd 02    Mask and restore X lower.
1222  1317  lpb 17
1223  4002  std 02
1224  2003  ldd 03    Store exponent in cell 01.
1225  4001  std 01
1226  2325  ldb 25    Modify count to equalize delay.
1227  0701  sbn 01

1230  6103  nzf 03    Zero check count.
1231  0300  nop       Time delay.
1232  6003  zjf 03
1233  1600  scc 00    Complement and restore count.
1234  7777  7777
1235  4334  stb 34
1236  0300  nop       Mask first bit of Y lower and add
1237  2004  ldd 04        X lower.

1240  1241  lpf 41
1241  3002  add 02    Jump back to over flow check routine.
1242  6572  nzb 72
1243  0423  ldn 23    Time delay.
1244  0300  nop
1245  0701  sbn 01
1246  6501  nzb 01
1247  2347  ldb 47    Check sign of answer.
```

```
1250  6105  nzf 05
1251  0404  ldn 04      Answer is positive. Time delay and
1252  0701  sbn 01          return jump to main program.
1253  6501  nzb 01
1254  6005  zjf 05
1255  2401  lcd 01      Answer is negative. Complement and
1256  4001  std 01          restore cells 02 and 01. Return
1257  2402  lcd 02          jump to main program.

1260  4002  std 02
1261  7007  jpi 07
1262  2200  ldc 00      X is much larger than Y. Jump to d
1263  0164  0164            delay routine.
1264  6517  nzb 17
1265  2003  ldd 03      Y is much larger than X. Load Y in
1266  4001  std 01          cells 01 and 02 and jump to
1267  2004  ldd 04          delay routine.

1270  4002  std 02
1271  2200  ldc 00
1272  0157  0157
1273  6527  nzb 27
1274  2002  ldd 02      Exponents are equal. Right shift
1275  0111  ls 6            X and Y lower one place and
1276  0110  ls 3            add. Store result in cell 02.
1277  0103  ls 2

1300  1200  lpc 00
1301  3777  3777
1302  4002  std 02
1303  2004  ldd 04
1304  0111  ls 6
1305  0110  ls 3
1306  0103  ls 2
1307  1306  lpb 06

1310  5002  rad 02
1311  5401  aod 01      Modify exponent.
1312  2200  ldc 00      Jump to delay routine.
1313  0152  0152
1314  6550  nzb 50
1315  0400  ldn 00      zeroize temp. storage        BEGIN  FMU
1316  4261  stf 61
1317  4261  stf 61

1320  4261  stf 61
1321  4261  stf 61
1322  2401  lcd 01      load first argument (upper)
1323  6307  njf 07      test for neg or positive
1324  4001  std 01      store positive arg
1325  2402  lcd 02      compl and store lower
1326  4002  std 02
1327  0401  ldn 01      remember first number neg
```

```
1330  4251  stf 51
1331  6206  pjf 06      orig number pos - delay
1332  2200  ldc 00
1333  0005  0005
1334  0701  sbn 01
1335  6501  nzb 01
1336  0300  nop
1337  2002  ldd 02      zero test first arg

1340  6106  nzf 06
1341  2200  ldc 00
1342  0477  0477
1343  0701  sbn 01
1344  6501  nzb 01
1345  7007  jpi 07
1346  2403  lcd 03      repeat tests on second arg
1347  6307  njf 07

1350  4003  std 03
1351  2404  lcd 04
1352  4004  std 04
1353  0501  lcn 01
1354  5225  raf 25
1355  6206  pjf 06
1356  2200  ldc 00
1357  0005  0005

1360  0701  sbn 01
1361  6501  nzb 01
1362  0300  nop
1363  2004  ldd 04
1364  6111  nzf 11
1365  4001  std 01
1366  4002  std 02
1367  2200  ldc 00

1370  0462  0462
1371  0701  sbn 01
1372  6501  nzb 01
1373  0300  nop
1374  7007  jpi 07
1375  0514  lcn 14
1376  6305  njf 05      jump over temp storage
1377  0000

1400  0000
1401  0000
1402  0000
1403  4301  stb 01      store bit count (-14)
1404  4705  srb 05      shift lower ans left one
1405  2305  ldb 05      load upper ans
1406  6207  pjf 07      check need for carry to upper
1407  1200  lpc 00      remove left bit from lower
```

```
1410  3777  3777
1411  4311  stb 11      place into lower
1412  5713  aob 13      execute carry
1413  0401  ldn 01      set up and jump back
1414  6205  pjf 05
1415  2200  ldc 00      delay
1416  0004  0004
1417  0701  sbn 01

1420  6501  nzb 01
1421  4721  srb 21
1422  2004  ldd 04      begin next bit multiplication
1423  6220  pjf 20      if next bit zero, no multiplication this loop
1424  2002  ldd 02
1425  5325  rab 25      add to lower ans
1426  1402  lsd 02      check for occurrance of end around carry
1427  0201  lpn 01

1430  6006  zjf 06      jump if no corr for carry
1431  5732  aob 32      corr for carry
1432  0501  lcn 01
1433  5333  rab 33
1434  0401  ldn 01
1435  6212  pjf 12
1436  2200  ldc 00      delay
1437  0003  0003

1440  0701  sbn 01
1441  6501  nzb 01
1442  6205  pjf 05
1443  2200  ldc 00      delay
1444  0010  0010
1445  0701  sbn 01
1446  6501  nzb 01
1447  4404  srd 04

1450  5746  aob 46
1451  6745  njb 45      return for next bit mult; if no jump, coef mult compl
1452  2351  ldb 51      restore sign record for later ref
1453  4203  stf 03
1454  6203  pjf 03
1455  6302  njf 02
1456  0000
1457  2003  ldd 03      add exponents

1460  3600  sbc 00
1461  2000  2000
1462  5001  rad 01
1463  6346  njf 46
1464  2365  ldb 65      adjusts for zero coef
1465  6107  nzf 07
1466  0501  lcn 01      reduce exp by one
1467  5001  rad 01
```

```
1470  6341  njf 41
1471  0500  lcn 00      zeroize coef
1472  4002  std 02
1473  6307  njf 07
1474  4002  std 02
1475  2200  ldc 00      delay
1476  0002  0002
1477  0701  sbn 01

1500  6501  nzb 01
1501  0300  nop
1502  2002  ldd 02
1503  6305  njf 05
1504  4402  srd 02      left justify coef and adj exp
1505  0501  lcn 01
1506  5001  rad 01
1507  6205  pjf 05

1510  2200  ldc 00
1511  0003  0003
1512  0701  sbn 01
1513  6501  nzb 01
1514  2336  ldb 36      examine sign of ans
1515  6006  zjf 06
1516  2401  lcd 01      complement ans upper and lower if sign is neg
1517  4001  std 01

1520  2402  lcd 02
1521  4002  std 02
1522  6206  pjf 06
1523  2200  ldc 00
1524  0004  0004
1525  0701  sbn 01
1526  6501  nzb 01
1527  0300  nop

1530  7007  jpi 07      return to M.P.
1531  0501  lcn 01
1532  7777  hlt 77
1533  0400  ldn 00      zeroize temp storage                BEGIN  FDV
1534  4261  stf 61
1535  4261  stf 61
1536  2403  lcd 03      test arguments as for FMU EXCEPT DIVISOR =0
1537  6307  njf 07           gives error halt

1540  4003  std 03
1541  2404  lcd 04
1542  4004  std 04
1543  0401  ldn 01
1544  4252  stf 52
1545  6206  pjf 06
1546  2200  ldc 00
1547  0005  0005
```

```
1550  0701  sbn 01
1551  6501  nzb 01
1552  0300  nop
1553  2004  ldd 04
1554  6036  zjf 36
1555  2401  lcd 01
1556  6307  njf 07
1557  4001  std 01

1560  2402  lcd 02
1561  4002  std 02
1562  0501  lcn 01
1563  5233  raf 33
1564  6206  pjf 06
1565  2200  ldc 00
1566  0005  0005
1567  0701  sbn 01

1570  6501  nzb 01
1571  0300  nop
1572  2002  ldd 02
1573  6110  nzf 10
1574  4001  std 01
1575  2200  ldc 00
1576  0242  0242
1577  0701  sbn 01

1600  6501  nzb 01
1601  0300  nop
1602  7007  jpi 07
1603  0514  lcn 14      set counter
1604  4210  stf 10
1605  2002  ldd 02      coef check for start
1606  3404  sbd 04
1607  6210  pjf 10      coef 04 greater than 02 jump to 1617

1610  5403  aod 03
1611  6142  nzf 42      jump to 1653
1612  0500  lcn 00
1613  7772  hlt 72
1614  0000
1615  0000
1616  0000
1617  2200  ldc 00      delay

1620  0002  0002
1621  0701  sbn 01
1622  6501  nzb 01
1623  2002  ldd 02
1624  3404  sbd 04
1625  6305  njf 05      jump to 1632
1626  4002  std 02
1627  5712  aob 12      in 1615
```

```
1630  0401  ldn 01
1631  6206  pjf 06
1632  2200  ldc 00      delay
1633  0002  0002
1634  0701  sbn 01
1635  6501  nzb 01
1636  0300  nop
1637  5723  aob 23      increase counter by one

1640  6220  pjf 20      coef div compl; jump to 1660
1641  4724  srb 24      on 1615
1642  2002  ldd 02
1643  6310  njf 10      jump to 1653
1644  4402  srd 02
1645  6721  njb 21      jump to 1624
1646  2200  ldc 00
1647  0004  0004

1650  0701  sbn 01
1651  6501  nzb 01
1652  6613  pjb 13      jump to 1637
1653  4402  srd 02
1654  3404  sbd 04
1655  4002  std 02
1656  0401  ldn 01
1657  6630  pjb 30

1660  2200  ldf 00      form exponent
1661  2001  ldd 01
1662  3403  sbd 03
1663  5001  rad 01
1664  6752  njb 52
1665  2350  ldb 50      place coef of ans in 0002
1666  4002  std 02
1667  2351  ldb 51      check for sign of ans

1670  6107  nzf 07
1671  2200  ldc 00
1672  0003  0003
1673  0701  sbn 01
1674  6501  nzb 01
1675  0300  nop
1676  6005  zjf 05
1677  2401  lcd 01      compl for neg ans

1700  4001  std 01
1701  2402  lcd 02
1702  4002  std 02
1703  7007  jpi 07      return jump to M.P.
```

APPENDIX II

STATE GENERATOR AND TEST

1. Theory.

Linear approximation between samples, as follows:

$$\dot{\theta} = \frac{\theta B - \theta A}{\Delta t} \qquad \text{A II-1}$$

Both θB and θA are the average of four samples taken as quickly as possible (141.3 u sec/sample). The time, Δt, between θB and θA is 840 cycle times, or at 6.4 second/cycle time, 5,376 u sec.

2. Memory requirements.

Lower memory: 01-04, 07, 10, 20-47, and 61-67.

Load State Generator: 2000-2322.

Load Output Section: 2323-2357.

3. Subroutines required.

Constant Time Floating Point Arithmetic Package. (APP. I)

A quick "Divide-by-4" routine, included with this appendix.

4. The programs for the state generator and the "Divide-by-4" subroutine follow. Comments are included with the programs.

STATE GENERATOR

```
1761  7500  exf 00
1762  1401  1401
1763  7600  ina 00
1764  4262  stf 62
1765  7500  exf 00
1766  1401  1401
1767  7600  ina 00
1770  4257  stf 57

1771  7500  exf 00
1772  1401  1401
1773  7600  ina 00
1774  4254  stf 54
1775  7500  exf 00
1776  1401  1401
1777  7600  ina 00
2000  4251  stf 51

2001  4001  std 01
2002  0101  pta
2003  0604  adn 04
2004  4007  std 07
2005  7061  jpi 61
2006  2001  ldd 01
2007  4021  std 21
2010  2240  ldf 40

2011  4001  std 01
2012  0101  pta
2013  0604  adn 04
2014  4007  std 07
2015  7061  jpi 61
2016  2001  ldd 01
2017  5021  rad 21
2020  2227  ldf 27

2021  4001  std 01
2022  0101  pta
2023  0604  adn 04
2024  4007  std 07
2025  7061  jpi 61
2026  2001  ldd 01
2027  5021  rad 21
2030  2216  ldf 16
```

```
2031  4001  std 01
2032  0101  pta
2033  0604  adn 04
2034  4007  std 07
2035  7061  jpi 61
2036  2001  ldd 01
2037  5021  rad 21      0A  avg in 0021
2040  2600  lcf 00

2041  0464  ldn 64
2042  0601  adn 01
2043  6501  nzb 01
2044  0300  nop
2045  6205  pjf 05
2046  0A1   0000
2047  0A2   0000
2050  0A3   0000

2051  0A4   0000
2052  7500  exf 00
2053  1401  1401
2054  7600  ina 00
2055  4256  stf 56
2056  7500  exf 00
2057  1401  1401
2060  7600  ina 00

2061  4253  stf 53
2062  7500  exf 00
2063  1401  1401
2064  7600  ina 00
2065  4250  stf 50
2066  7500  exf 00
2067  1401  1401
2070  7600  ina 00

2071  4245  stf 45
2072  4001  std 01
2073  0101  pta
2074  0604  adn 04
2075  4007  std 07
2076  7061  jpi 61
2077  2001  ldd 01
2100  4022  std 22

2101  2234  ldf 34
2102  4001  std 01
2103  0101  pta
2104  0604  adn 04
2105  4007  std 07
2106  7061  jpi 61
2107  2001  ldd 01
2110  5022  rad 22
```

```
2111  2223  ldf 23
2112  4001  std 01
2113  0101  pta
2114  0604  adn 04
2115  4007  std 07
2116  7061  jpi 61
2117  2001  ldd 01
2120  5022  rad 22

2121  2212  ldf 12
2122  4001  std 01
2123  0101  pta
2124  0604  adn 04
2125  4007  std 07
2126  7061  jpi 61
2127  2001  ldd 01
2130  5022  rad 22      θB  avg in 0022

2131  6206  pjf 06
2132  6306  njf 06
2133  0B1   0000
2134  0B2   0000
2135  0B3   0000
2136  0B4   0000
2137  0300  nop
2140  3421  sbd 21

2141  4024  std 24      θB - θA
2142  0101  pta
2143  0605  adn 05
2144  4007  std 07
2145  2024  ldd 24
2146  7066  jpi 66      θB - θA to floating point
2147  2001  ldd 01
2150  4030  std 30

2151  2002  ldd 02
2152  4031  std 31
2153  2044  ldd 44
2154  4003  std 03
2155  2045  ldd 45
2156  4004  std 04
2157  0101  pta
2160  0604  adn 04

2161  4007  std 07
2162  7065  jpi 65      θB DOT formed
2163  2001  ldd 01
2164  4032  std 32
2165  2002  ldd 02
2166  4033  std 33
2167  7500  exf 00
2170  1401  1401
```

```
2171  7600  ina 00
2172  4256  stf 56
2173  7500  exf 00
2174  1401  1401
2175  7600  ina 00
2176  4253  stf 53
2177  7500  exf 00
2200  1401  1401

2201  7600  ina 00
2202  4250  stf 50
2203  7500  exf 00
2204  1401  1401
2205  7600  ina 00
2206  4245  stf 45
2207  4001  std 01
2210  0101  pta

2211  0604  adn 04
2212  4007  std 07
2213  7061  jpi 61
2214  2001  ldd 01
2215  4023  std 23
2216  2234  ldf 34
2217  4001  std 01
2220  0101  pta

2221  0604  adn 04
2222  4007  std 07
2223  7061  jpi 61
2224  2001  ldd 01
2225  5023  rad 23
2226  2223  ldf 23
2227  4001  std 01
2230  0101  pta

2231  0604  adn 04
2232  4007  std 07
2233  7061  jpi 61
2234  2001  ldd 01
2235  5023  rad 23
2236  2212  ldf 12
2237  4001  std 01
2240  0101  pta

2241  0604  adn 04
2242  4007  std 07
2243  7061  jpi 61
2244  2001  ldd 01
2245  5023  rad 23      ΘC avg in 0023
2246  6206  pjf 06
2247  6306  njf 06
2250  0C1   0000
```

```
2251  0C2   0000
2252  0C3   0000
2253  0C4   0000
2254  0300  nop
2255  3422  sbd 22      ΘC - ΘB formed
2256  4025  std 25
2257  0101  pta
2260  0605  adn 05

2261  4007  std 07
2262  2025  ldd 25
2263  7066  jpi 66      ΘC - ΘB to floating point
2264  2001  ldd 01
2265  4034  std 34
2266  2002  ldd 02
2267  4035  std 35
2270  2044  ldd 44

2271  4003  std 03
2272  2045  ldd 45
2273  4004  std 04
2274  0101  pta
2275  0604  adn 04
2276  4007  std 07
2277  7065  jpi 65      ΘC DOT formed
2300  2001  ldd 01

2301  4036  std 36
2302  2002  ldd 02
2303  4037  std 37
```

---

continue to OUTPUT ROUTINE or CONTROL PROGRAM.

SUBROUTINE DIVIDE BY FOUR

```
1704  2001  ldd 01     pick-up argument
1705  6212  pjf 12     sign check
1706  0111  ls 6   )
1707  0110  ls 3   }   divide by 4
1710  0102  ls 1   )

1711  1200  lpc 00   }  mask
1712  1777  1777     )
1713  3200  adc 00      extend negative sign bit
1714  6000  6000
1715  4001  std 01     store answer
1716  6312  njf 12     jump by positive path
1717  0111  ls 6   )
1720  0110  ls 3   }   divide by 4
1721  0102  ls 1   )
1722  1200  lpc 00   }  mask
1723  1777  1777     )
1724  4001  std 01     store answer
1725  0300  nop    )
1726  0300  nop    }   time delay
1727  0300  nop    )
1730  7007  jpi 07     return jump to M.P.
```

Jump into and return jump out of this subroutine in the same manner as for the floating point routines.

OUTPUT ROUTINE FOR θ DOT

```
2353  2036  ldd 36
2354  4001  std 01
2355  2037  ldd 37
2356  4002  std 02
2357  2200  ldc 00

2360  2002  2002
2361  4003  std 03
2362  2200  ldc 00
2363  6221  6221
2364  4004  std 04
2365  0101  pta
2366  0604  adn 04
2367  4007  std 07

2370  7065  jpi 65      divide out w; normalizes output
2371  0101  pta
2372  0604  adn 04
2373  4007  std 07
2374  7067  jpi 67      θ DOT to fixed point
2375  0300  nop
2376  4020  std 20
2377  7500  exf 00

2400  2401  2401
2401  7304  out 04
2402  0021  0021
2403  6103  nzf 03
2404  6002  zjf 02
2405  0020  0020
2406  7010  jpi 10
2407  7757  hlt 57
```